Open text files

```
ifstream inFile( "test.txt", ios::in );
```

Open an existing file; if the file doesn't exit, fail

```
ofstream outFile( "test.txt", ios::out );
```

Create a file; if the file have existed, the data in it will be deleted.

```
ofstream outFile( "test.txt", ios::app );
```

Create a file; if the file have existed, the data in it will be keep

```
fstream ioFile( "test.txt", ios::in | ios::out );
```

Open an existing file; if the file doesn't exit, fail

Open binary files

```
ifstream inFile( "test.txt", ios::binary );
```

Open an existing file; if the file doesn't exit, fail

```
ofstream outFile( "test.txt", ios::binary );
```

Create a file; if the file have existed, the data in it will be deleted.

```
ofstream outFile( "test.txt", ios::app | ios::binary );
```

Create a file; if the file have existed, the data in it will be keep

```
fstream ioFile( "test.txt", ios::in | ios::out | ios::binary );
```

Open an existing file; if the file doesn't exit, fail

Open binary files

```
ifstream inFile;
inFile.open( "test.dat", ios::binary );

ofstream outFile;
outFile.open( "test.dat", ios::binary );

ofstream outFile;
outFile.open( "test.dat", ios::app | ios::binary );

fstream ioFile;
ioFile.open( "test.dat", ios::in | ios::out | ios::binary );
```

Open an existing file;
if the file doesn't exit, fail

```
ifstream inFile( "test.txt", ios::in );

ifstream inFile( "test.dat", ios::binary );

fstream ioFile( "test.txt", ios::in | ios::out );

fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );
```

Create a file; if the file have existed, the data in it will be deleted

```
ofstream outFile( "test.txt", ios::out );

ofstream outFile( "test.txt", ios::binary );
```

Create a file;
if the file have existed,
the data in it will be keep

```
ofstream outFile( "test.txt", ios::app );

ofstream outFile( "test.txt", ios::app | ios::binary );
```

Load data from a text file

```
char name[ 4 ];
int calculus;

inFile >> name >> calculus;


char buf[ 9 ];
inFile.get( buf, sizeof( buf ), '\n' );
inFile.getline( buf, sizeof( buf ), '\n' );


char ch;
inFile.get( ch );
```

Load data from a text file

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

Grade grade;
inFile >> grade.name >> grade.calculus;


char buf[ 9 ];
inFile.get( buf, sizeof( buf ), '\n' );
inFile.getline( buf, sizeof( buf ), '\n' );


char ch;
inFile.get( ch );
```

Load data from a binary file

```
char name[ 4 ];
inFile.read( name, 4 );

inFile.read( name, sizeof( name ) );
```

Load data from a binary file

```
int calculus;

inFile.read( calculus, 4 );

inFile.read( calculus, sizeof( calculus ) );
```

Wrong!

Load data from a binary file

```
int calculus;

inFile.read( reinterpret_cast< char * >( &calculus ), 4 );

inFile.read( reinterpret_cast< char * >( &calculus ), sizeof( calculus ) );
```

Load data from a binary file

```
char name[ 4 ];
int calculus;
inFile.read( name, sizeof( name ) );
inFile.read( reinterpret_cast< char * >( &calculus ), sizeof( calculus ) );


struct Grade
{
   char name[ 4 ];
   int calculus;
};
Grade grade;
inFile.read( reinterpret_cast< char * >( &grade ), sizeof( grade ) );
```

The Prototype of read and write

```
inFile.read( char *s, int n );

outFile.write( const char *s, int n );
```

reinterpret_cast

```
void fun( char *p );

int main()
{
   char name[ 4 ] = "aaa";
   fun( name );
}

void fun( char *p )
{
   cout << p << endl;
}
```

reinterpret_cast

```
void fun( char *p );

int main()
{
   int calculus = 100;
   fun( &calculus );     Wrong!
}

void fun( char *p )
{
   cout << p << endl;
}
```

reinterpret_cast

```
void fun( char *p );

int main()
{
   int calculus = 100;
   fun( reinterpret_cast< char * >( & calculus ) );
}

void fun( char *p )
{
   cout << p << endl;
}
```

reinterpret_cast

```
void fun( char *p );

int main()
{
   int calculus = 100;
   fun( reinterpret_cast< const char * >( &calculus ) );
}

void fun( const char *p )
{
   cout << p << endl;
}
```

reinterpret_cast

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

void fun( char *p );

int main()
{
   Grade grade = { "aaa", 100 };
   fun( &grade );   Wrong!
}

void fun( char *p )
{
   cout << p << endl;
}
```

reinterpret_cast

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

void fun( char *p );

int main()
{
   Grade grade = { "aaa", 100 };
   fun( reinterpret_cast< char * >( &grade ) );
}

void fun( char *p )
{
   cout << p << endl;
}
```

reinterpret_cast

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

void fun( char *p );

int main()
{
   Grade grade = { "aaa", 100 };
   fun( reinterpret_cast< const char * >( &grade ) );
}

void fun( const char *p )
{
   cout << p << endl;
}
```

Save data to text file

```
char name[ 4 ];
int calculus;

outFile << name << calculus;


char ch;
inFile.put( ch );
```

Save data to text file

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

Grade grade = { "aaa", 100 };

outFile << grade.name << grade.calculus;


char ch;

inFile.put( ch );
```

Save data to binary file

```
char name[ 4 ] = "aaa";
outFile.write( name, 4 );

outFile.write( name, sizeof( name ) );
```

Save data to binary file

```
int calculus = 100;
outFile.write( calculus, 4 );

outFile.write( calculus, sizeof( calculus ) );
```

Wrong!

Save data to binary file

```
int calculus = 100;
outFile.write( reinterpret_cast< const char * >( &calculus ), 4 );

outFile.write( reinterpret_cast< const char * >( &calculus ),
               sizeof( calculus ) );
```

Save data to binary file

```
char name[ 4 ] = "aaa";

int calculus = 100;

outFile.write( name, sizeof( name ) );

outFile.write( reinterpret_cast< const char * >( &calculus ),

               sizeof( calculus ) );


struct Grade
{
   char name[ 4 ];
   int calculus;
};

Grade grade = { "aaa", 100 };

outFile.write( reinterpret_cast< const char * >( &grade ),

               sizeof( grade ) );
```

Move the file position pointer

```
inFile.seekg( 10, ios::beg );

inFile.seekg( 10, ios::cur );

inFile.seekg( 10, ios::end );

outFile.seekp( 10, ios::beg );

outFile.seekp( 10, ios::cur );

outFile.seekp( 10, ios::end );
```

Return the value of the file position pointer

```
inFile.tellg();

outFile.tellp();
```

Set iostream to good state

```
inFile.clear();

outFile.clear();
```

Read from and write to a binary file

```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

const char name[ 4 ] = "aaa";

ioFile.write( name, 4 );

ioFile.seekg( 0, ios::beg );

char buffer[ 4 ];

ioFile.read( buffer, 4 );

buffer[ 4 ] = '\0';

cout << buffer << endl;

ioFile.close();
```

Read from and write to a binary file

```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

const int calculus = 100;

ioFile.write( calculus, 4 );     Wrong!

ioFile.seekg( 0, ios::beg );

int number;

ioFile.read( number, 4 );     Wrong!

cout << number << endl;

ioFile.close();
```

Read from and write to a binary file

```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

const int calculus = 100;

ioFile.write( reinterpret_cast< const char * > ( &calculus ), 4 );

ioFile.seekg( 0, ios::beg );

int number;

ioFile.read( reinterpret_cast< char * > ( &number ), 4 );

cout << number << endl;

ioFile.close();
```

Read from and write to a binary file

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

Grade grade1 = { "aaa", 100 };

ioFile.write( reinterpret_cast< const char * >( &grade1 ), 8 );

ioFile.seekg( 0, ios::beg );

Grade grade2;

ioFile.read( reinterpret_cast< char * > ( &grade2 ), 8 );

cout << grade2.id << endl << grade2.calculus << endl;

ioFile.close();
```

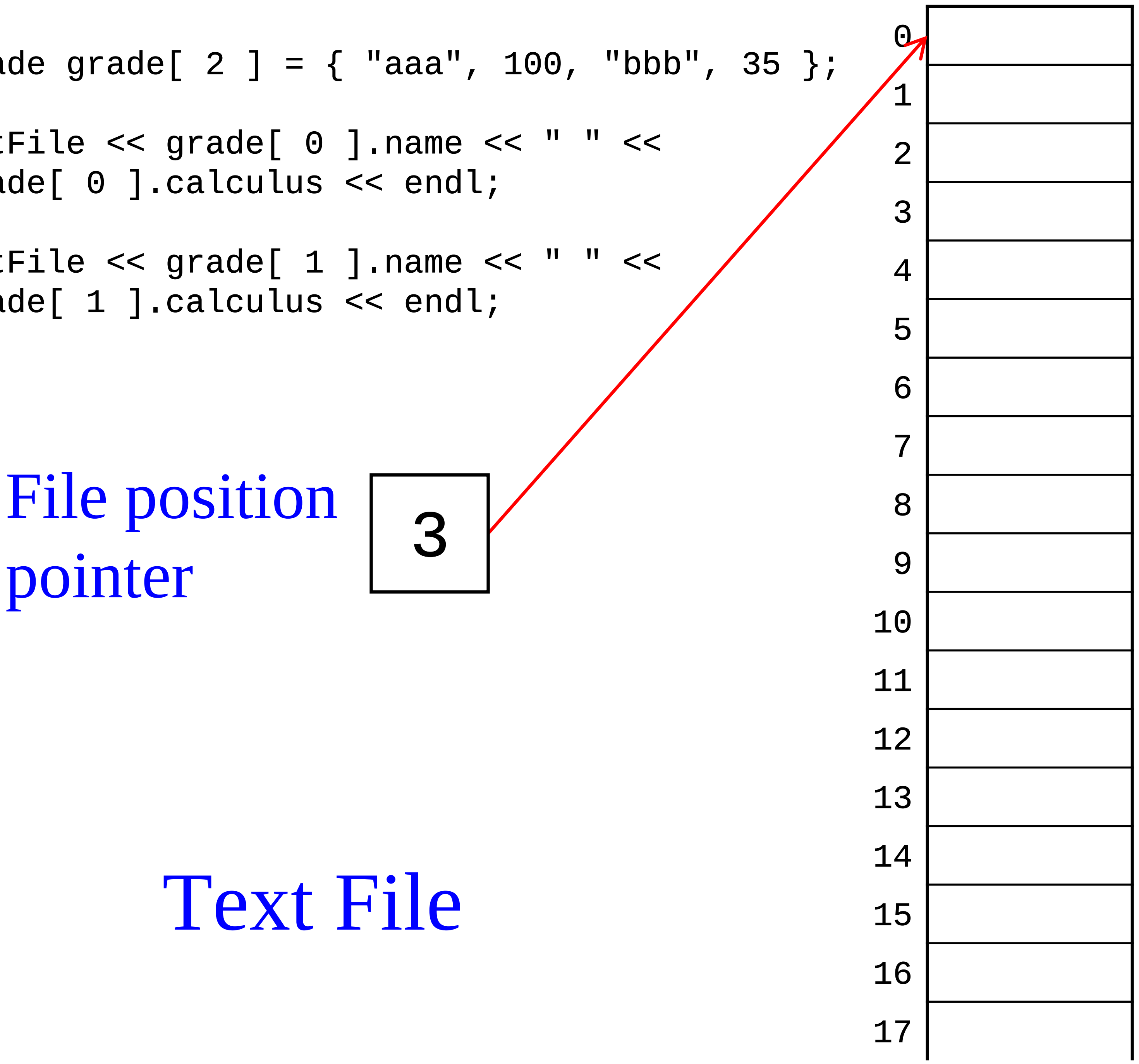
Read from and write to a binary file

```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

Grade grade1 = { "aaa", 100 };

ioFile.write( reinterpret_cast< const char * > ( &grade1 ),
              sizeof( Grade ) );

ioFile.seekg( 0, ios::beg );

Grade grade2;

ioFile.read( reinterpret_cast< char * > ( &grade2 ),
sizeof( Grade ) );

cout << grade2.id << endl << grade2.calculus << endl;

ioFile.close();
```

Text file

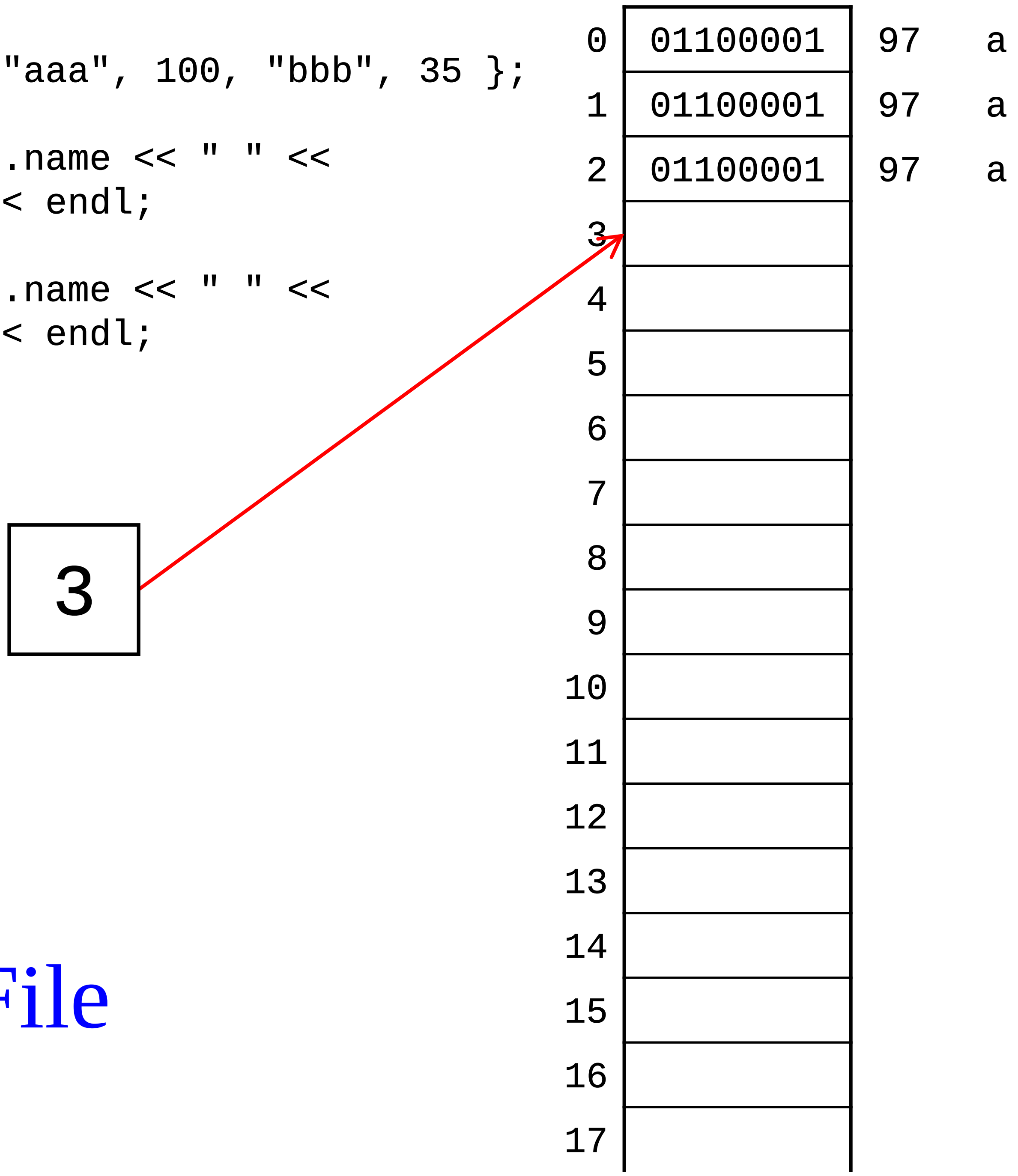
```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

ofstream outFile( "Grade.txt", ios::out );

outFile << grade[ 0 ].name << " " << grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " << grade[ 1 ].calculus << endl;
```

```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```



Text File

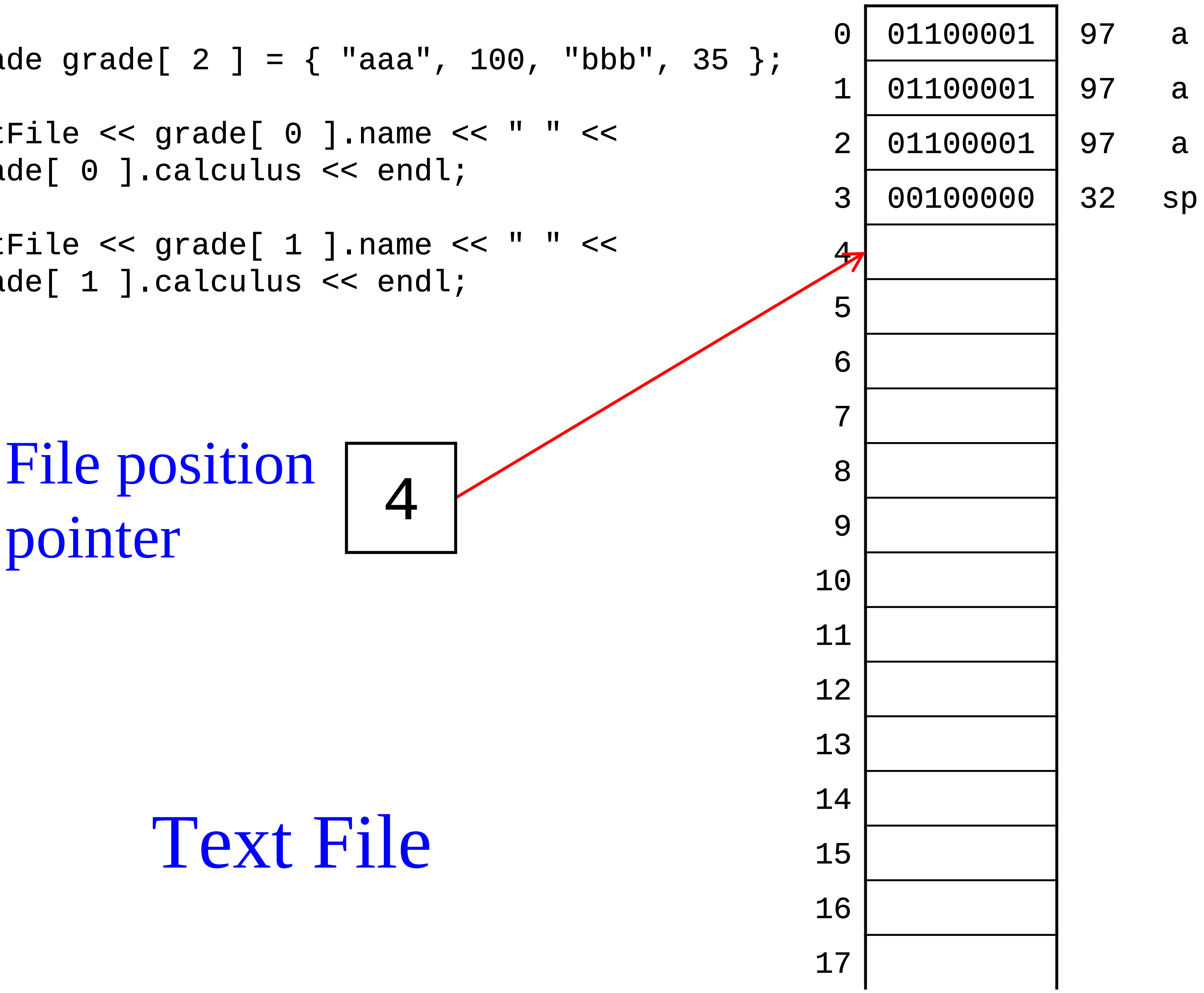
```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```



File position pointer

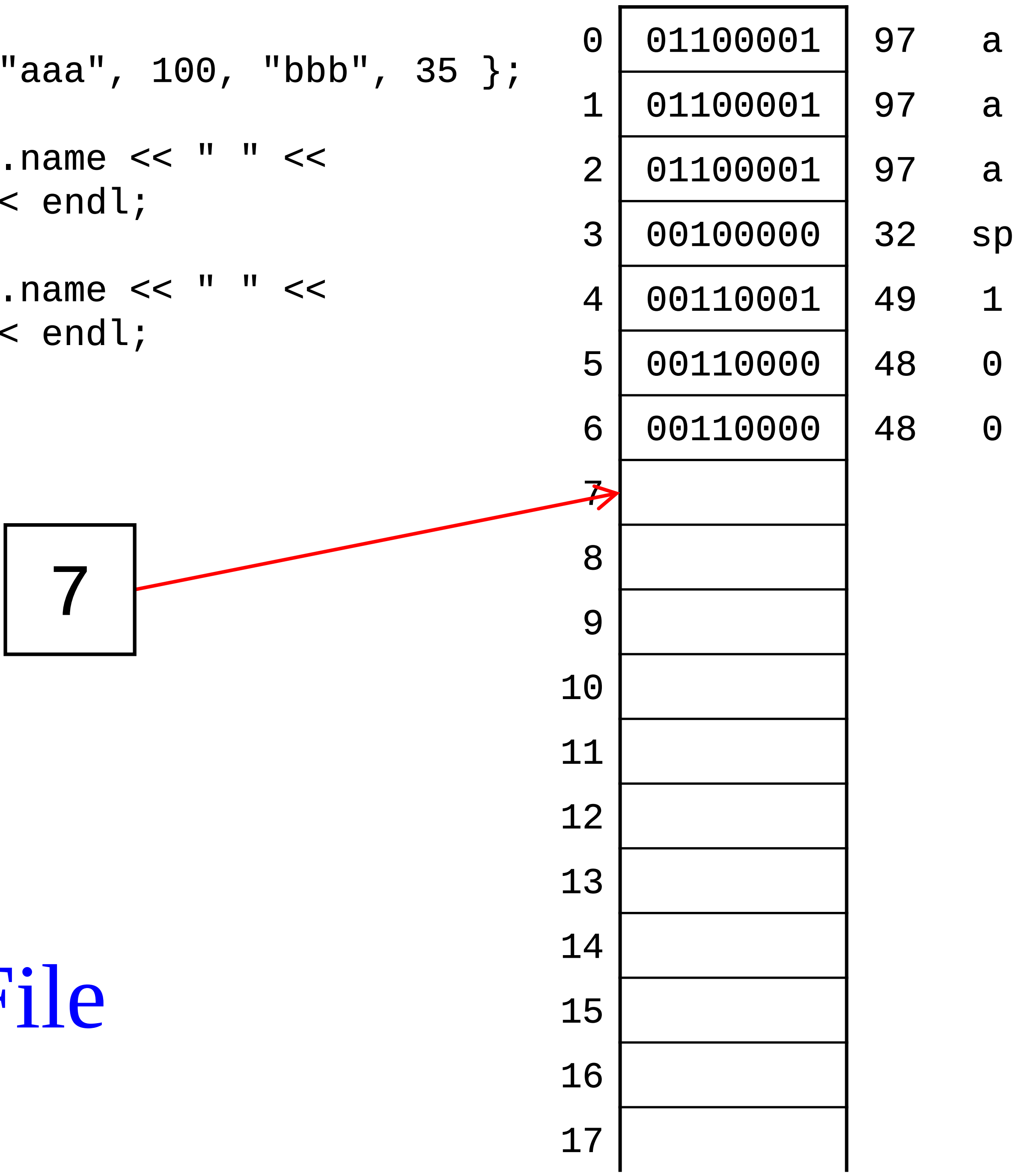
Text File

```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```



Text File

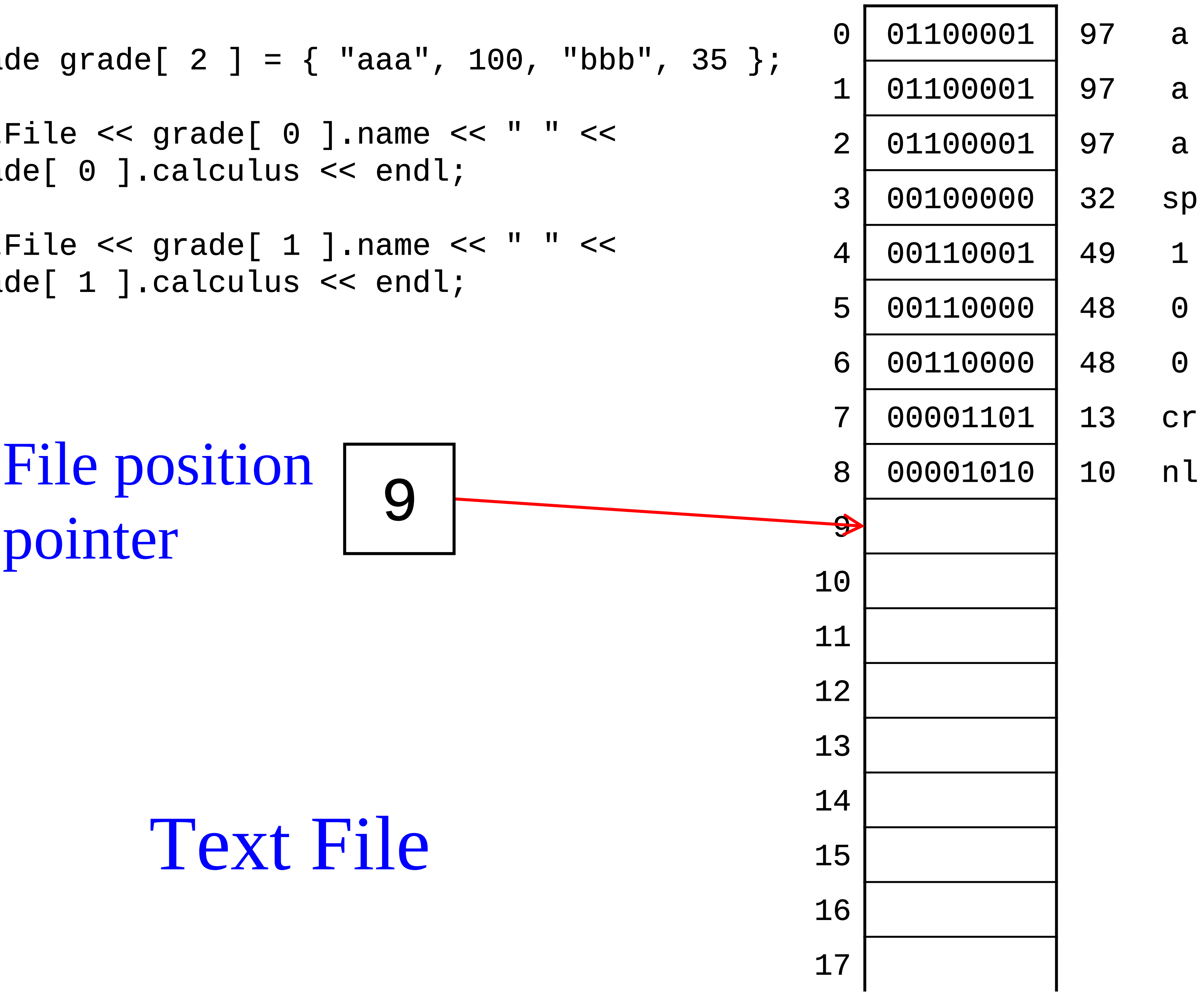
```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```



0	01100001	97	a
1	01100001	97	a
2	01100001	97	a
3	00100000	32	sp
4	00110001	49	1
5	00110000	48	0
6	00110000	48	0
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			

Text File

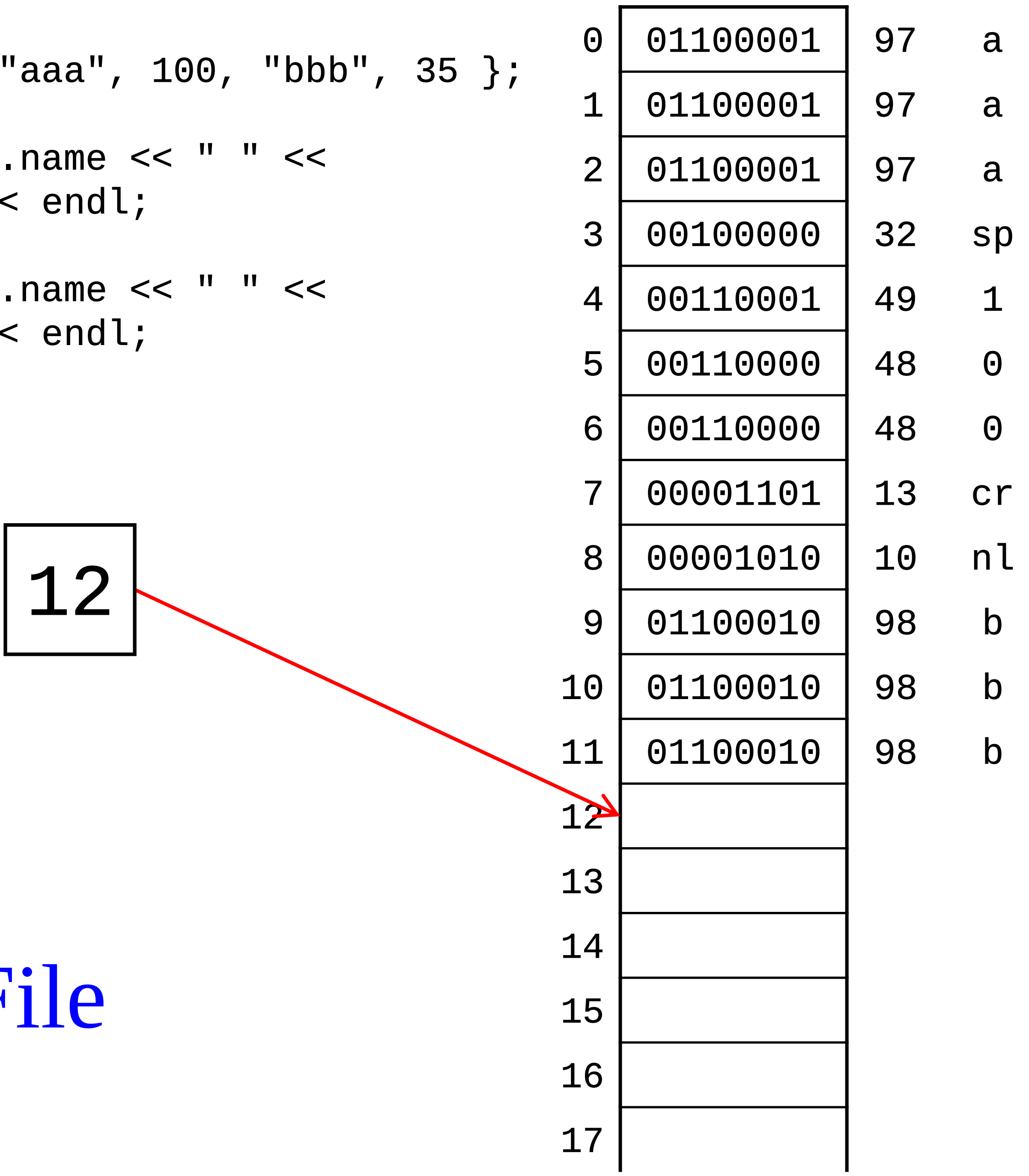
```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```



Text File

```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```

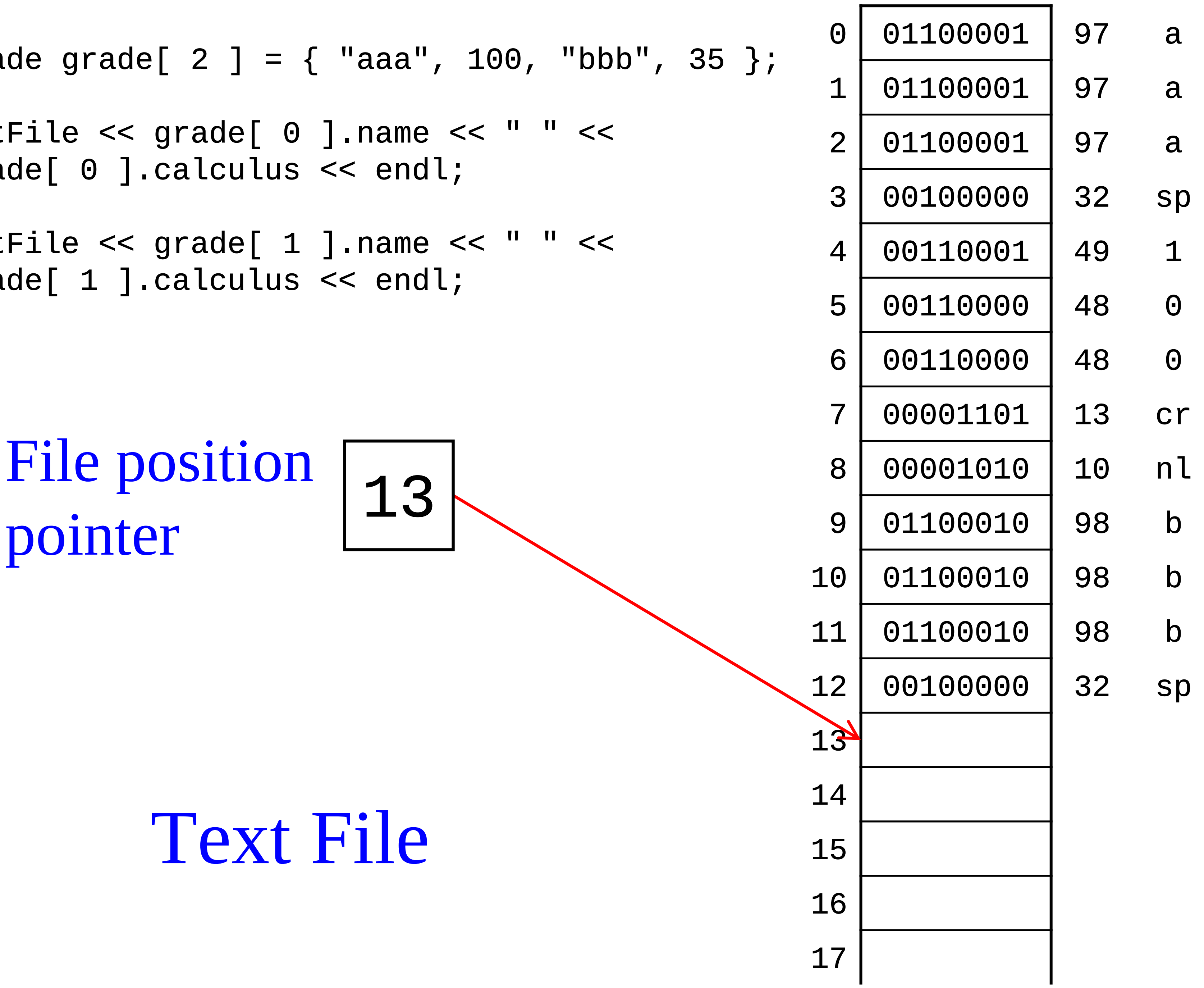
File position pointer



0	01100001	97	a
1	01100001	97	a
2	01100001	97	a
3	00100000	32	sp
4	00110001	49	1
5	00110000	48	0
6	00110000	48	0
7	00001101	13	cr
8	00001010	10	nl
9	01100010	98	b
10	01100010	98	b
11	01100010	98	b
12			
13			
14			
15			
16			
17			

Text File

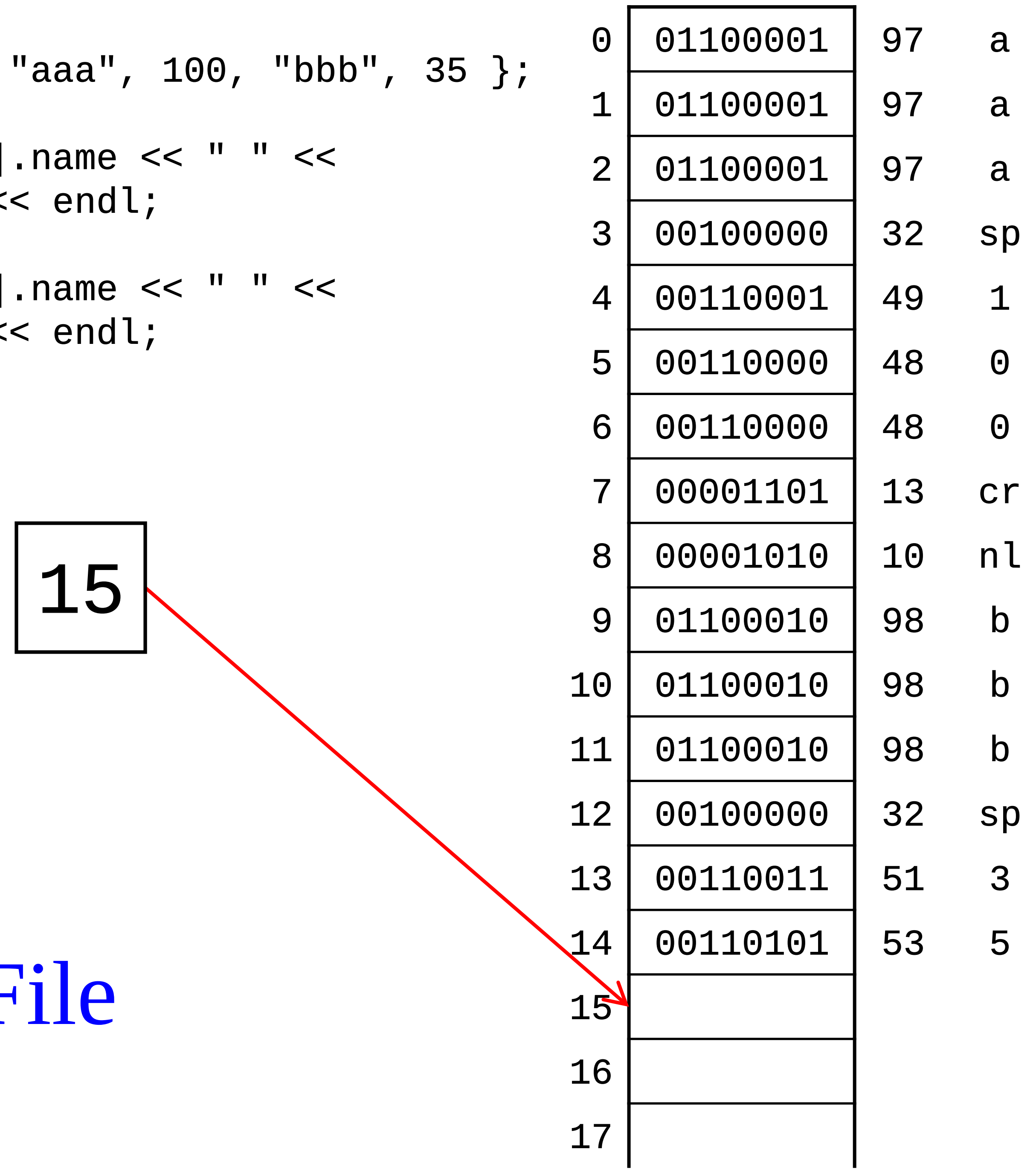
```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```



0	01100001	97	a
1	01100001	97	a
2	01100001	97	a
3	00100000	32	sp
4	00110001	49	1
5	00110000	48	0
6	00110000	48	0
7	00001101	13	cr
8	00001010	10	nl
9	01100010	98	b
10	01100010	98	b
11	01100010	98	b
12	00100000	32	sp
13			
14			
15			
16			
17			

Text File

```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```

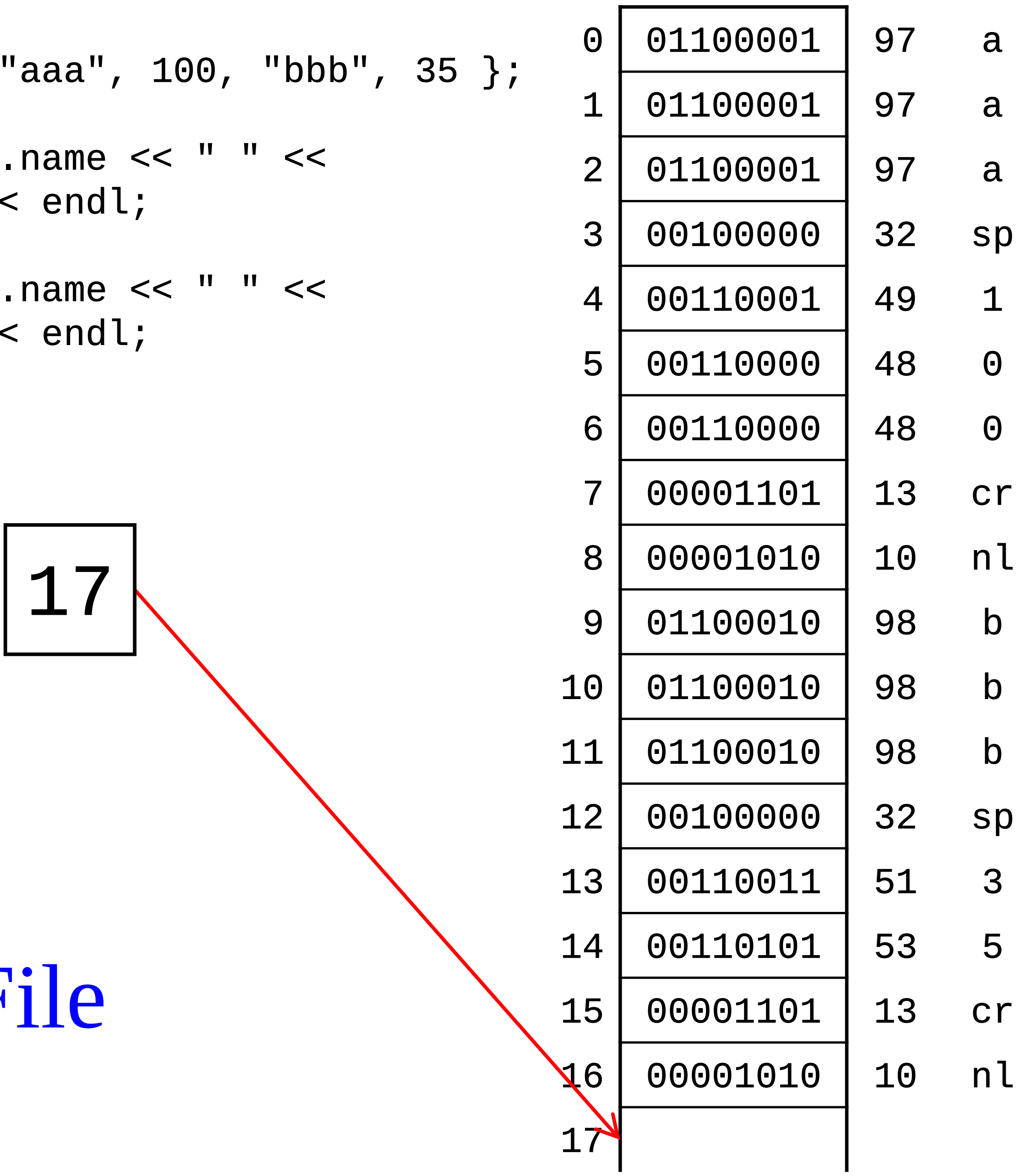


0	01100001	97	a
1	01100001	97	a
2	01100001	97	a
3	00100000	32	sp
4	00110001	49	1
5	00110000	48	0
6	00110000	48	0
7	00001101	13	cr
8	00001010	10	nl
9	01100010	98	b
10	01100010	98	b
11	01100010	98	b
12	00100000	32	sp
13	00110011	51	3
14	00110101	53	5
15			
16			
17			

Text File

```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

outFile << grade[ 0 ].name << " " <<
grade[ 0 ].calculus << endl;

outFile << grade[ 1 ].name << " " <<
grade[ 1 ].calculus << endl;
```

File position pointer



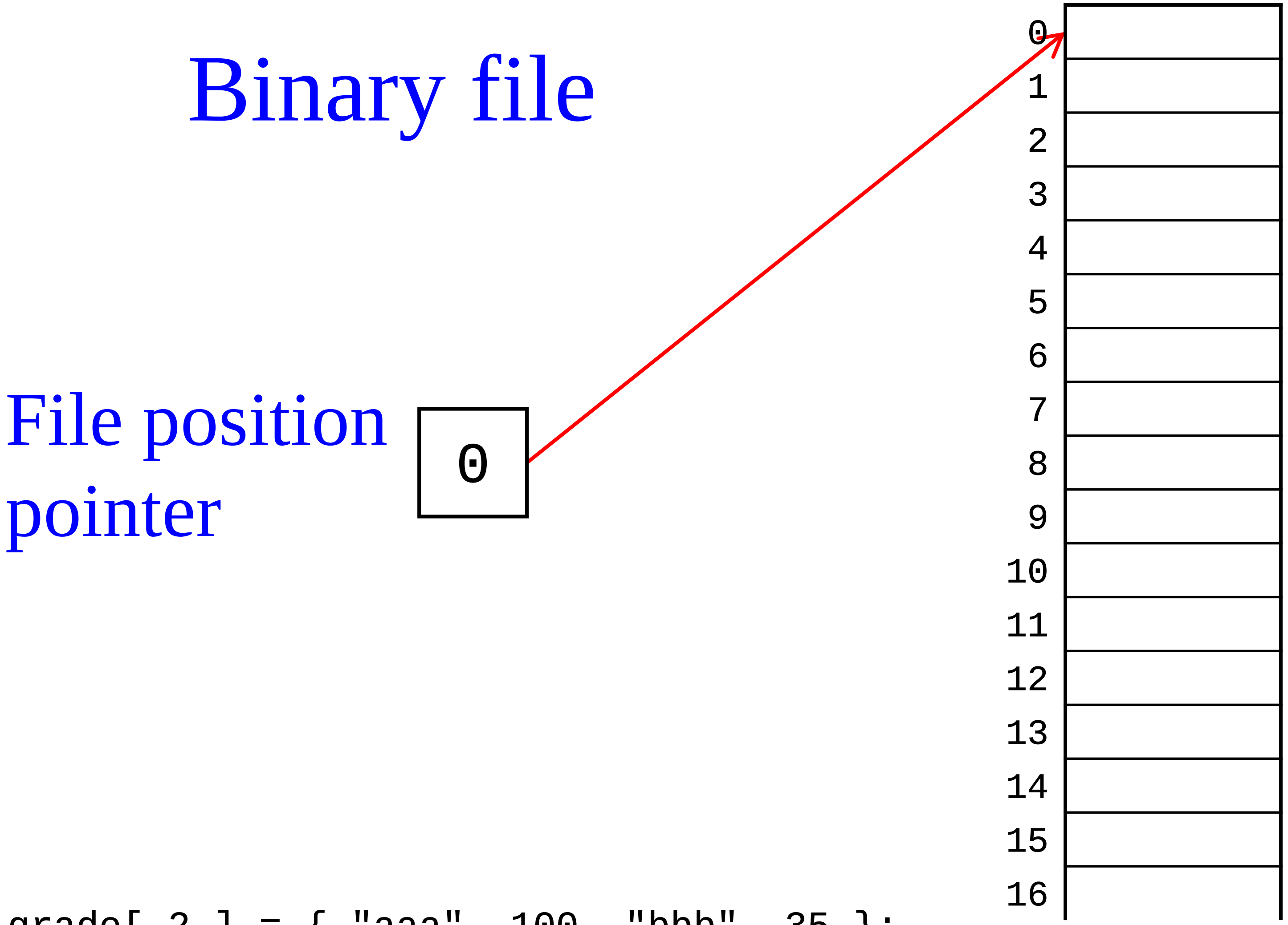
0	01100001	97	a
1	01100001	97	a
2	01100001	97	a
3	00100000	32	sp
4	00110001	49	1
5	00110000	48	0
6	00110000	48	0
7	00001101	13	cr
8	00001010	10	nl
9	01100010	98	b
10	01100010	98	b
11	01100010	98	b
12	00100000	32	sp
13	00110011	51	3
14	00110101	53	5
15	00001101	13	cr
16	00001010	10	nl
17			

Text File

Binary file

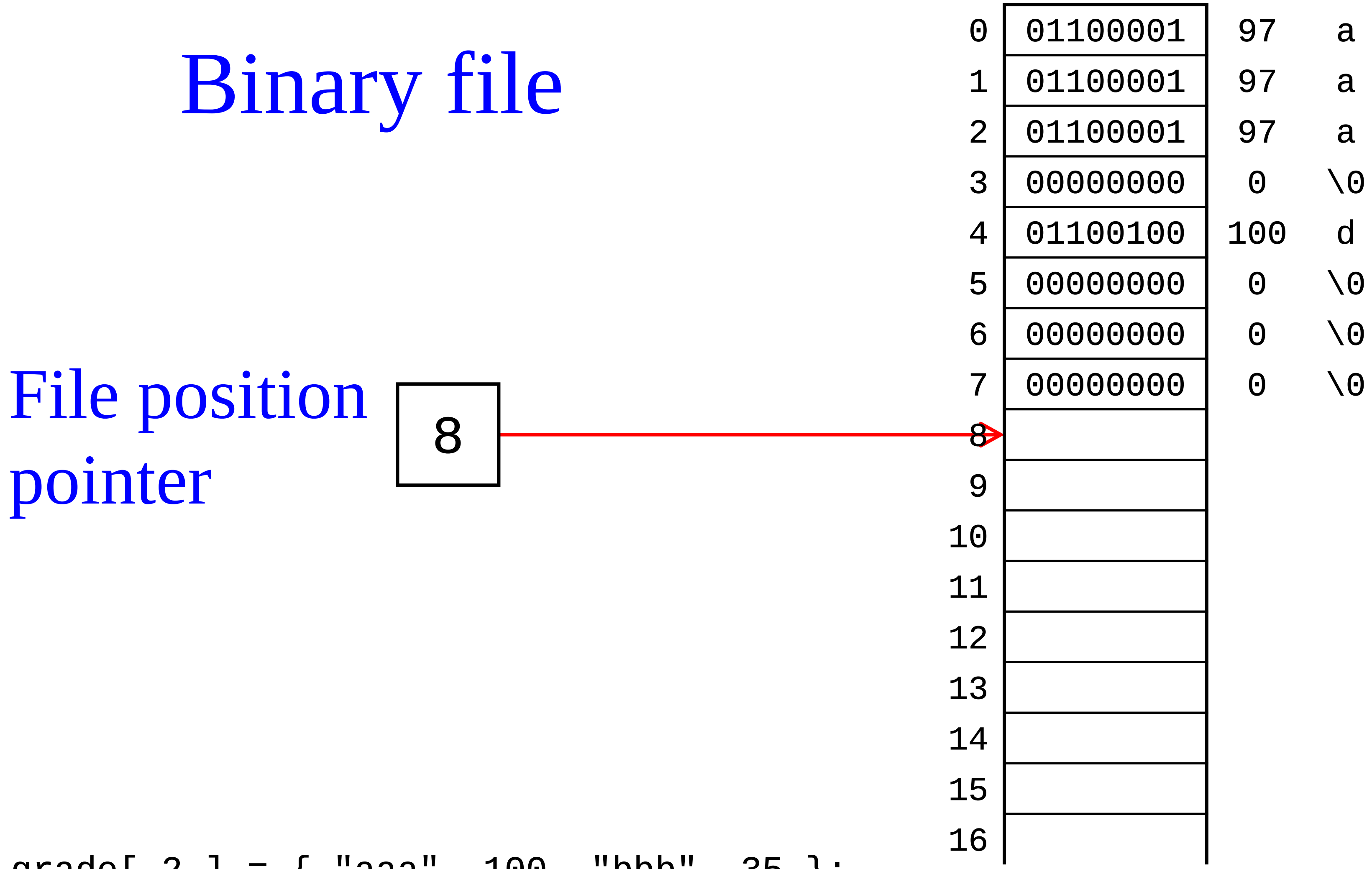
```
struct Grade
{
   char name[ 4 ];
   int calculus;
};

Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };

ofstream outFile( "Grade.dat", ios::binary );

outFile.write( reinterpret_cast< const char * >( &grade[ 0 ] ) ),
               sizeof( Grade ) );

outFile.write( reinterpret_cast< const char * >( &grade[ 1 ] ) ),
               sizeof( Grade ) );
```

Binary file



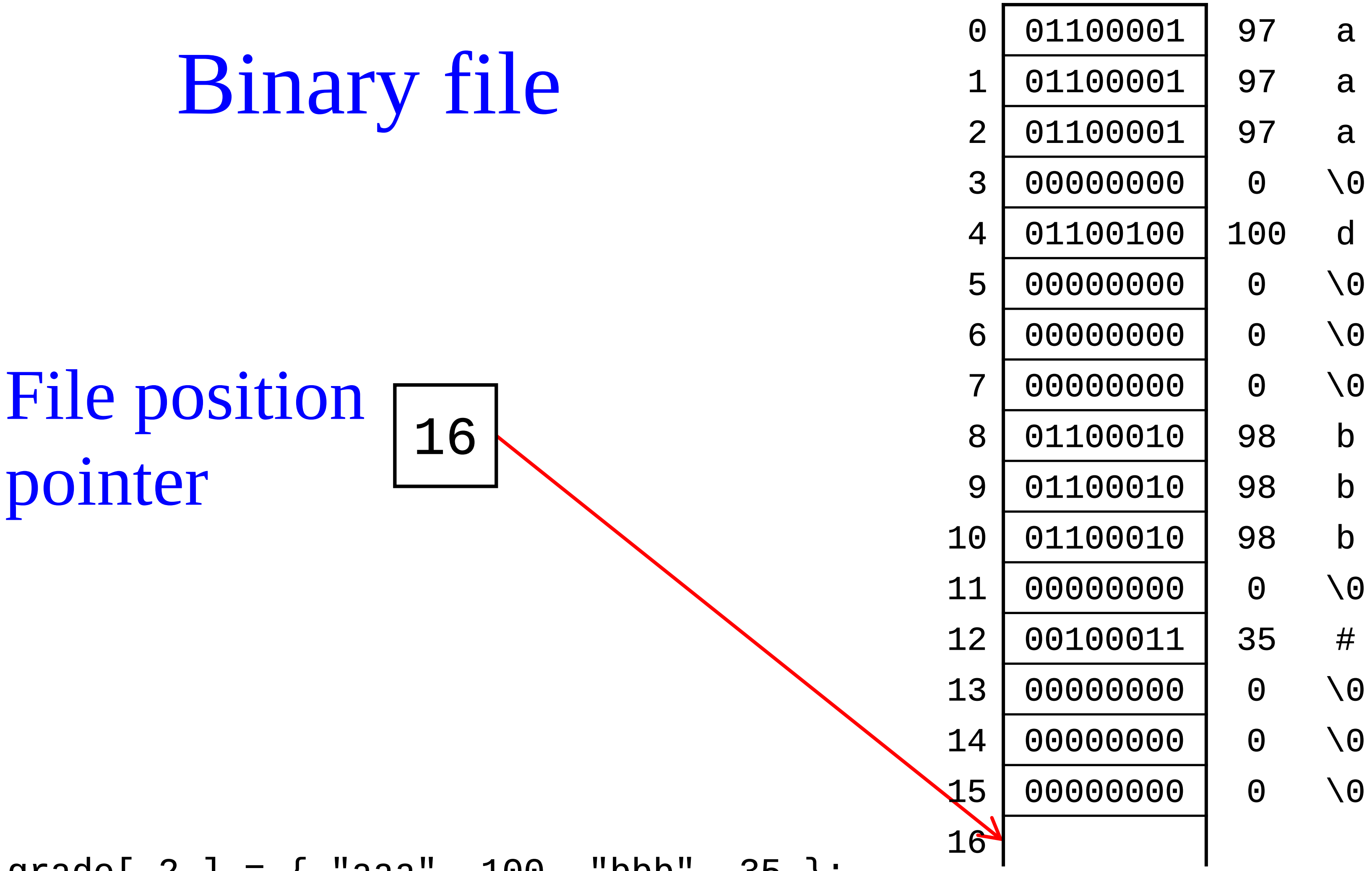
```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };
outFile.write( reinterpret_cast< const char * >( &grade[ 0 ] ) ),
               sizeof( Grade );
outFile.write( reinterpret_cast< const char * >( &grade[ 1 ] ) ),
               sizeof( Grade );
```

Binary file



```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };
outFile.write( reinterpret_cast< const char * >( &grade[ 0 ] ) ),
               sizeof( Grade );
outFile.write( reinterpret_cast< const char * >( &grade[ 1 ] ) ),
               sizeof( Grade );
```

Binary file



0	01100001	97	a
1	01100001	97	a
2	01100001	97	a
3	00000000	0	\0
4	01100100	100	d
5	00000000	0	\0
6	00000000	0	\0
7	00000000	0	\0
8	01100010	98	b
9	01100010	98	b
10	01100010	98	b
11	00000000	0	\0
12	00100011	35	#
13	00000000	0	\0
14	00000000	0	\0
15	00000000	0	\0
16			

```
Grade grade[ 2 ] = { "aaa", 100, "bbb", 35 };
outFile.write( reinterpret_cast< const char * >( &grade[ 0 ] ),
               sizeof( Grade ) );
outFile.write( reinterpret_cast< const char * >( &grade[ 1 ] ),
               sizeof( Grade ) );
```

Input all records from a binary file

Structure

```
struct Grade
{
   char name[ 8 ];
   int calculus;
};
```

```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

Grade grade[ 3 ] = { "aaa", 100, "bbb", 35, "ccc", 69 };

for( int i = 0; i <= 2; i++ )
   ioFile.write( reinterpret_cast< const char * > ( &grade[ i ] ),
                 sizeof( Grade ) );

ioFile.seekp( 0, ios::beg );

Grade points[ 3 ];

int k = -1;

while( !ioFile.eof() )
{
   k++;
   ioFile.read( reinterpret_cast< char * > ( &points[ k ] ),
                sizeof( Grade ) );
}
```

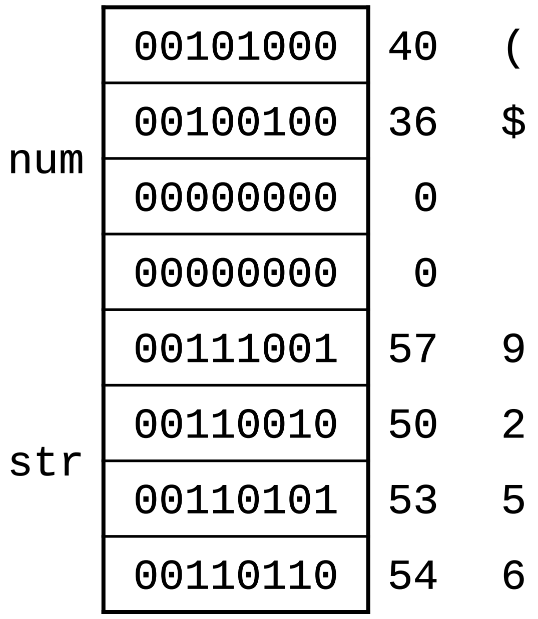
```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

Grade grade[ 3 ] = { "aaa", 100, "bbb", 35, "ccc", 69 };

for( int i = 0; i <= 2; i++ )
   ioFile.write( reinterpret_cast< const char * > ( &grade[ i ] ),
                 sizeof( Grade ) );

ioFile.seekp( 0, ios::beg );

Grade points[ 3 ];

int k = 0;

while( ioFile.read( reinterpret_cast< char * >( &points[ k ] ),
                    sizeof( Grade ) ) )
   k++;
```

```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

Grade grade[ 3 ] = { "aaa", 100, "bbb", 35, "ccc", 69 };

int i;

for( i = 0; i <= 2; i++ )
   ioFile.write( reinterpret_cast< const char * > ( &grade[ i ] ),
                 sizeof( Grade ) );

Grade points[ 3 ];




ioFile.seekg( 0, ios::beg );

for( i = 0; i <= recordNumber; i++ )
   ioFile.read( reinterpret_cast< char * > ( &points[ i ] ),
                sizeof( Grade ) );
```

```
fstream ioFile( "test.dat", ios::in | ios::out | ios::binary );

Grade grade[ 3 ] = { "aaa", 100, "bbb", 35, "ccc", 69 };

int i;

for( i = 0; i <= 2; i++ )
   ioFile.write( reinterpret_cast< const char * > ( &grade[ i ] ),
                 sizeof( Grade ) );

Grade points[ 3 ];

ioFile.seekg( 0, ios::end );

int recordNumber = ioFile.tellg() / sizeof( Grade );

ioFile.seekg( 0, ios::beg );

for( i = 0; i <= recordNumber; i++ )
   ioFile.read( reinterpret_cast< char * > ( &points[ i ] ),
                sizeof( Grade ) );
```

```c
char str[] = "9256";
unsigned num = 9256;
```

In memory



8 + 32 + 1024 + 8192 = 9256

$(00000000\ 00000000\ 00100100\ 00101000)_2 = (9256)_{10}$

ASCII character set										
	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	'	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

```
char str[] = "9256";
unsigned num = 9256;
```

0	00101000	40	(
1	00100100	36	$
2	00000000	0	
3	00000000	0	
4	00111001	57	9
5	00110010	50	2
6	00110101	53	5
7	00110110	54	6

In binary file

```
outFile.write( reinterpret_cast< const char * > ( &num ), 4 );
outFile.write( str, 4 );
```

```
char str[] = "9256";
unsigned num = 9256;
```

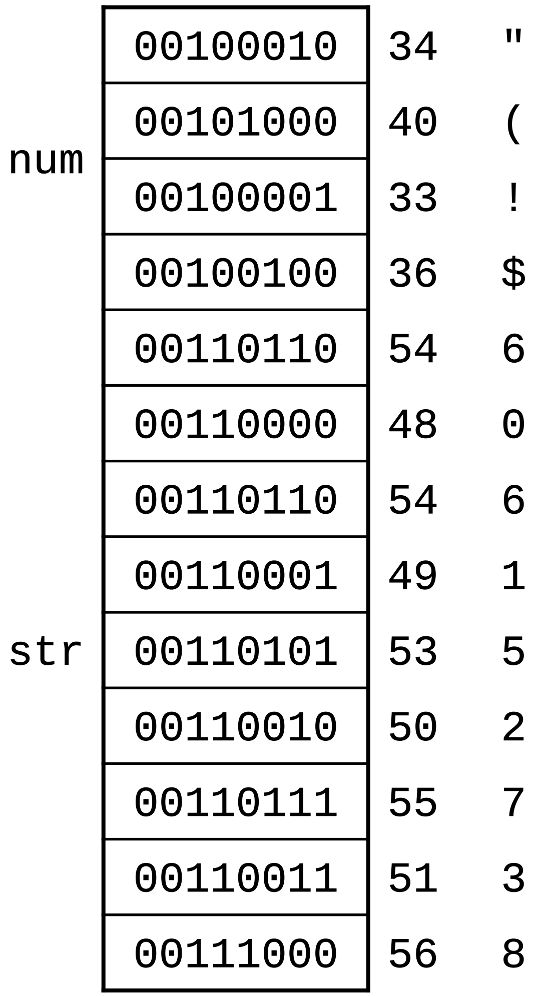
In text file

0	00111001	57	9
1	00110010	50	2
2	00110101	53	5
3	00110110	54	6
4	00111001	57	9
5	00110010	50	2
6	00110101	53	5
7	00110110	54	6

```
outFile << num << str;
```

```c
char str[] = "606152738";
unsigned num = 606152738;
```

In memory



	00100010	34	"
	00101000	40	(
num	00100001	33	!
	00100100	36	$
	00110110	54	6
	00110000	48	0
	00110110	54	6
	00110001	49	1
str	00110101	53	5
	00110010	50	2
	00110111	55	7
	00110011	51	3
	00111000	56	8

2 + 32 + 2048 + 8192 + 65536 + 2097152 + 67108864 + 536870912
= 606152738

$(00100100\ 00100001\ 00101000\ 00100010)_2 = (606152738)_{10}$

```
char str[] = "606152738";
unsigned num = 606152738;
```

In binary file

0	00100010	34	"
1	00101000	40	(
2	00100001	33	!
3	00100100	36	$
4	00110110	54	6
5	00110000	48	0
6	00110110	54	6
7	00110001	49	1
8	00110101	53	5
9	00110010	50	2
10	00110111	55	7
11	00110011	51	3
12	00111000	56	8

```
outFile.write( reinterpret_cast< const char * > ( &num ), 4 );
outFile.write( str, 9 );
```

```
char str[] = "606152738";
unsigned num = 606152738;
```

In text file

```
outFile << num << str;
```

0	00110110	54	6
1	00110000	48	0
2	00110110	54	6
3	00110001	49	1
4	00110101	53	5
5	00110010	50	2
6	00110111	55	7
7	00110011	51	3
8	00111000	56	8
9	00110110	54	6
10	00110000	48	0
11	00110110	54	6
12	00110001	49	1
13	00110101	53	5
14	00110010	50	2
15	00110111	55	7
16	00110011	51	3
17	00111000	56	8